Assignment 06 — Genomic Prediction

PUBH 8878

Requirements

- Show complete work for derivations or justifications.
- Submit well-documented R code with clear comments and a fixed seed (`set.seed(8878)` unless otherwise noted).
- Interpret results in a breeding/genetics context, not just numbers.
- Submit the rendered PDF (or HTML/Docx if requested); do not submit the source `.qmd`.

Setup (packages)

```r
# You may need: install.packages(c("BGLR","sommer","glmnet","ggplot2","dplyr","tidyr","tibbl
library(BGLR)
library(glmnet)
```

```
Loading required package: Matrix

Loaded glmnet 4.1-10
```

```r
library(ggplot2)
library(dplyr)
```

```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(tidyr)
```

```

Attaching package: 'tidyr'

The following objects are masked from 'package:Matrix':

    expand, pack, unpack
```

```r
library(tibble)
# Optional (Problem 2):
# library(sommer)
set.seed(8878)
```

Data and helper utilities (provided)

```r
# Load BGLR wheat data
data(wheat)
if (exists("wheat") && is.list(wheat)) {
  X <- wheat$X; Y <- wheat$Y; A <- wheat$A
} else {
  X <- wheat.X; Y <- wheat.Y; A <- wheat.A
}

stopifnot(nrow(X) == nrow(Y), nrow(A) == nrow(X))
n <- nrow(X); m <- ncol(X); e <- ncol(Y)

# Build genomic relationship matrix G (center by allele freq; scale by variance)
build_G <- function(X){
  p <- colMeans(X)
  W <- sweep(X, 2, p, "-")
  Den <- sum(p * (1 - p))
  G <- tcrossprod(W) / Den
  G
}

G <- build_G(X)
```

```r
# Normalize A and G to comparable scale
A <- A / mean(diag(A))
G <- G / mean(diag(G))

# Line-wise folds (outer CV)
make_folds <- function(n, K=5, seed=8878){ set.seed(seed); sample(rep(seq_len(K), length.out

# Metrics
metrics <- function(obs, pred){
  c(Accuracy = suppressWarnings(cor(obs, pred)), RMSE = sqrt(mean((obs - pred)^2)))
}

# Ridge on X with inner CV for lambda
cv_ridge_X <- function(y, X, fold, inner_nfolds=5){
  acc <- rmse <- lambda <- numeric(max(fold))
  for(k in seq_len(max(fold))){
    tr <- fold != k; te <- !tr
    fit <- cv.glmnet(X[tr, , drop=FALSE], y[tr], alpha = 0, standardize = TRUE,
                     nfolds = inner_nfolds, intercept = TRUE)
    pred <- as.numeric(predict(fit, newx = X[te, , drop=FALSE], s = "lambda.min"))
    mm <- metrics(y[te], pred)
    acc[k] <- mm["Accuracy"]; rmse[k] <- mm["RMSE"]; lambda[k] <- fit$lambda.min
  }
  list(acc = acc, rmse = rmse, lambda = lambda,
       acc_mean = mean(acc), acc_sd = sd(acc), rmse_mean = mean(rmse))
}

# Kernel ridge via BLUP (BGLR) for a given K (A or G)
cv_bglr_kernel <- function(y, K, fold, nIter=6000, burnIn=1000){
  acc <- rmse <- numeric(max(fold))
  for(k in seq_len(max(fold))){
    ytr <- y; ytr[fold == k] <- NA
    fit <- BGLR(y = ytr, ETA = list(list(K = K, model = "RKHS")),
                nIter = nIter, burnIn = burnIn, verbose = FALSE)
    pred <- fit$yHat[fold == k]
    mm <- metrics(y[fold == k], pred)
    acc[k] <- mm["Accuracy"]; rmse[k] <- mm["RMSE"]
  }
  list(acc = acc, rmse = rmse,
       acc_mean = mean(acc), acc_sd = sd(acc), rmse_mean = mean(rmse))
}
```

```r
# A+G grid of weights
cv_bglr_AplusG <- function(y, A, G, fold, weights = seq(0,1,by=0.5)){
  grid <- expand.grid(wA = weights, wG = weights)
  grid <- grid[abs(grid$wA + grid$wG - 1) < 1e-8, , drop = FALSE]
  out <- lapply(seq_len(nrow(grid)), function(i){
    wA <- grid$wA[i]; wG <- grid$wG[i]
    Kmix <- wA*A + wG*G
    res <- cv_bglr_kernel(y, Kmix, fold)
    c(wA=wA, wG=wG, acc_mean=res$acc_mean, acc_sd=res$acc_sd, rmse_mean=res$rmse_mean)
  })
  as.data.frame(do.call(rbind, out))
}
```

Problem 1 — Baseline across environments (20 pts)

Implement the line-wise K-fold CV pipeline for each environment separately (Env 1–4): Ridge(X), A-BLUP(A), GBLUP(G), and a light A+G grid with weights {(1,0), (0,1), (0.5,0.5)}. Report Accuracy (cor) ± SD and RMSE for each model–environment pair and include one bar plot per environment.

Guidance

- Use `K=5` outer folds via `make_folds(n, K=5, seed=8878)`.
- Tune Ridge lambda on the training portion only (the function above already does this).
- Keep A and G normalized (done above) and IDs aligned.

```r
K <- 5
fold <- make_folds(n, K)

run_env <- function(y){
  ridge <- cv_ridge_X(y, X, fold)
  ares  <- cv_bglr_kernel(y, A, fold)
  gres  <- cv_bglr_kernel(y, G, fold)
  ag    <- cv_bglr_AplusG(y, A, G, fold, weights = c(0, 0.5, 1))
  best  <- ag[which.max(ag$acc_mean), ]
  tibble(
    Model    = c("Ridge(X)", "A-BLUP (A)", "GBLUP (G)", sprintf("A+G (wA=%.1f,wG=%.1f)", best$
    Accuracy = c(ridge$acc_mean, ares$acc_mean, gres$acc_mean, best$acc_mean),
    `Acc SD` = c(ridge$acc_sd,   ares$acc_sd,   gres$acc_sd,   best$acc_sd),
    RMSE     = c(ridge$rmse_mean, ares$rmse_mean, gres$rmse_mean, best$rmse_mean)
  )
}
```

```r
env_summaries <- lapply(1:e, function(j) run_env(as.numeric(Y[, j])))
names(env_summaries) <- paste0("Env ", 1:e)
env_summaries

# Example plot for Env 1
env_summaries[["Env 1"]] %>%
  ggplot(aes(x = reorder(Model, Accuracy), y = Accuracy)) +
  geom_col() + coord_flip() +
  labs(x=NULL, y="Accuracy (cor)", title="Env 1: Line-wise 5-fold CV accuracy") +
  theme_minimal(base_size = 14)
```

Briefly interpret which kernels work best across environments and why (2–4 sentences).

Problem 2 — Multi-environment G×E (20 pts)

Fit the multi-environment mixed model with a genetic main effect and Line×Env interaction using `sommer::mmer`, comparing Gu = A vs Gu = G. Use line-wise folds: in each outer fold, mask all phenotypes for the held-out lines across all environments, fit the model on the rest, and predict the masked values. Report mean Accuracy across folds for each choice of Gu and discuss when G×E helps.

```r
library(sommer)

df <- data.frame(Line=factor(seq_len(n)), Env=factor(rep(1:e, each=n)), y=as.vector(Y))

cv_me <- function(df, Gu, K=5){
  foldL <- make_folds(nlevels(df$Line), K)
  levels(df$Line) <- as.character(seq_len(nlevels(df$Line)))
  lines <- levels(df$Line)
  acc <- numeric(K)
  for(k in seq_len(K)){
    test_lines <- lines[foldL==k]
    dcv <- df
    dcv$y[dcv$Line %in% test_lines] <- NA
    fit <- mmer(y ~ Env,
                random = ~ vsr(Line, Gu=Gu) + vsr(Line:Env, Gu=Gu),
                rcov = ~ units, data = dcv, verbose = FALSE)
    p <- predict(fit, classify="Env:Line")$pvals
    obs <- df[!is.na(df$y), c("Env","Line","y")]
    mg  <- merge(p[,c("Env","Line","predicted.value")], obs, by=c("Env","Line"))
    mg  <- mg[mg$Line %in% test_lines,]
    acc[k] <- suppressWarnings(cor(mg$y, mg$predicted.value))
  }
  mean(acc)
}

acc_me_A <- cv_me(df, A)
acc_me_G <- cv_me(df, G)
data.frame(Model=c("ME (A)", "ME (G)"), Accuracy=c(acc_me_A, acc_me_G))
```

Problem 3 — A+G weight tuning (15 pts)

Extend the A+G search to a denser grid (e.g., `seq(0,1,by=0.1)`) and perform the grid search inside each outer fold (i.e., select weights using only the training data in that fold). Summarize the weight stability across folds (e.g., table or heatmap of selected weights) and the resulting test accuracy.

```r
weights <- seq(0,1,by=0.1)
fold <- make_folds(n, 5)
y1 <- as.numeric(Y[,1])

select_w_per_fold <- function(y, A, G, fold, weights){
  K <- max(fold)
  chosen <- matrix(NA_real_, nrow=K, ncol=2, dimnames=list(paste0("Fold",1:K), c("wA","wG"))
  acc   <- numeric(K)
  for(k in seq_len(K)){
    tr <- fold != k; te <- !tr
    # inner search on training
    grid <- expand.grid(wA=weights, wG=weights)
    grid <- grid[abs(grid$wA + grid$wG - 1) < 1e-8, , drop=FALSE]
    best <- NULL; best_acc <- -Inf
    for(i in seq_len(nrow(grid))){
      Kmix <- grid$wA[i]*A + grid$wG[i]*G
      ytr <- y; ytr[!tr] <- NA
      fit <- BGLR(y=ytr, ETA=list(list(K=Kmix, model="RKHS")), nIter=4000, burnIn=800, verbos
      pred <- fit$yHat[te]
      ca <- suppressWarnings(cor(y[te], pred))
      if(!is.na(ca) && ca > best_acc){ best_acc <- ca; best <- grid[i, , drop=FALSE] }
    }
    chosen[k,] <- c(best$wA, best$wG)
    acc[k] <- best_acc
  }
  list(weights=as.data.frame(chosen), acc=acc)
}
```

```r
res_w <- select_w_per_fold(y1, A, G, fold, weights)
as_tibble(res_w$weights)
mean(res_w$acc)
```

Comment on the stability of selected weights and whether A+G provides a material gain over A or G alone in this dataset.

Problem 4 — Alternative CV designs (15 pts)

Evaluate an environment-wise holdout design (predicting a new environment): for each environment, hold it out entirely, fit on the remaining environments (using either A-BLUP or GBLUP), and predict the held-out env. Report accuracy per environment and compare to the line-wise design. Briefly justify which design better matches different breeding decisions.

```r
env_holdout_acc <- function(Y, Kmat){
  e <- ncol(Y); n <- nrow(Y)
  out <- numeric(e)
  for(j in seq_len(e)){
    y <- as.numeric(Y[,j])
    ytr <- y; ytr[] <- NA
    # train on remaining envs by averaging BLUPs from a model fit on stacked data
    df <- data.frame(Line=factor(seq_len(n)), Env=factor(rep(1:e, each=n)), y=as.vector(Y))
    # mask the target environment for all lines
    df$y[df$Env == j] <- NA
    library(sommer)
    fit <- mmer(y ~ Env, random = ~ vsr(Line, Gu=Kmat), rcov = ~ units, data = df, verbose =
    p <- predict(fit, classify="Env:Line")$pvals
    pj <- subset(p, Env==j)
    ord <- order(as.integer(as.character(pj$Line)))
    pred <- pj$predicted.value[ord]
    out[j] <- suppressWarnings(cor(y, pred))
  }
  out
}

acc_env_A <- env_holdout_acc(Y, A)
acc_env_G <- env_holdout_acc(Y, G)
rbind(A=acc_env_A, G=acc_env_G)
```

Problem 5 — Selection metrics and expected gain (15 pts)

Using your best single-environment model for Env 1, compute (i) Spearman rank correlation between predictions and observed yield, (ii) overlap of the top 10% predicted vs top 10%

observed lines, and (iii) expected gain, defined as the mean observed yield of the top 10% by prediction minus the overall mean. Comment on how these complement Pearson accuracy for selection decisions.

```r
topk_metrics <- function(obs, pred, k=0.10){
  n <- length(obs); kN <- max(1, floor(k*n))
  rk <- suppressWarnings(cor(obs, pred, method = "spearman"))
  ix_p <- order(pred, decreasing=TRUE)[1:kN]
  ix_o <- order(obs,  decreasing=TRUE)[1:kN]
  overlap <- length(intersect(ix_p, ix_o)) / kN
  gain <- mean(obs[ix_p]) - mean(obs)
  c(Spearman=rk, TopK_Overlap=overlap, Expected_Gain=gain)
}

# example usage with your best model predictions for Env 1
```

Problem 6 — Ridge vs GBLUP equivalence (10 pts)

On Env 1, standardize columns of `X` to mean 0 and unit variance. Show that predictions from Ridge(X) and GBLUP(G) are numerically close by reporting the maximum absolute difference across lines. Briefly describe the mapping between the ridge penalty and the mixed-model variance ratio and why predictions align.

```r
# Standardize X
Xs <- scale(X, center=TRUE, scale=TRUE)
y  <- as.numeric(Y[,1])
fold <- make_folds(n, 5)

# Ridge fit
fit_r <- cv.glmnet(Xs[fold!=1,], y[fold!=1], alpha=0, standardize=FALSE)
pred_r <- as.numeric(predict(fit_r, newx = Xs[fold==1,], s = "lambda.min"))

# GBLUP predictions for the same test set via kernel ridge on G
G2 <- tcrossprod(Xs) / ncol(Xs) # one simple scaling
G2 <- G2 / mean(diag(G2))
ytr <- y; ytr[fold==1] <- NA
fit_g <- BGLR(y=ytr, ETA=list(list(K=G2, model="RKHS")), nIter=4000, burnIn=800, verbose=FALS
pred_g <- fit_g$yHat[fold==1]

max_abs_diff <- max(abs(pred_r - pred_g))
max_abs_diff
```

Problem 7 — Sensitivity and reproducibility (5 pts)

Briefly assess sensitivity to seeds and MCMC settings: rerun your best Env-1 model under two additional seeds and a shorter BGLR chain (e.g., `nIter=3000, burnIn=600`). Summarize the variation in Accuracy and whether conclusions change.

Bonus (up to 5 pts)

- Fit elastic net or lasso on `X` for Env 1 and compare to Ridge/GBLUP. Discuss when sparsity might help or hurt for polygenic traits like yield.

Submission checklist

- Rendered PDF with clear figures and tables.
- Seeds reported; session info (R and package versions) included at the end.
- Brief narrative interpreting which models and CV designs best support different breeding decisions.

```r
sessionInfo()
```

```
R version 4.5.1 (2025-06-13)
Platform: aarch64-apple-darwin20
Running under: macOS Tahoe 26.0.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;

locale:
[1] C.UTF-8/C.UTF-8/C.UTF-8/C/C.UTF-8/C.UTF-8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices datasets  utils     methods   base

other attached packages:
[1] tibble_3.3.0  tidyr_1.3.1   dplyr_1.1.4   ggplot2_3.5.2 glmnet_4.1-10
[6] Matrix_1.7-3  BGLR_1.1.4

loaded via a namespace (and not attached):
 [1] gtable_0.3.6       jsonlite_2.0.0     compiler_4.5.1
 [4] BiocManager_1.30.26 renv_1.1.5         tidyselect_1.2.1
 [7] Rcpp_1.1.0         splines_4.5.1      scales_1.4.0
```

```
[10] yaml_2.3.10         fastmap_1.2.0       lattice_0.22-7
[13] R6_2.6.1            generics_0.1.4      shape_1.4.6.1
[16] knitr_1.50          iterators_1.0.14    MASS_7.3-65
[19] pillar_1.11.0       RColorBrewer_1.1-3  rlang_1.1.6
[22] xfun_0.52           truncnorm_1.0-9     cli_3.6.5
[25] withr_3.0.2         magrittr_2.0.3      digest_0.6.37
[28] foreach_1.5.2       grid_4.5.1          lifecycle_1.0.4
[31] vctrs_0.6.5         evaluate_1.0.3      glue_1.8.0
[34] farver_2.1.2        codetools_0.2-20    survival_3.8-3
[37] purrr_1.0.4         rmarkdown_2.29      pkgconfig_2.0.3
[40] tools_4.5.1         htmltools_0.5.8.1
```